[2018看雪] - 第二题 - Trie Tree

原创

Flying_Fatty 于 2018-07-20 13:14:56 发布 119 收藏
分类专栏： reverse

main中的结构很好分析

```
14  Scanf("%s", Input, 32);
15  sub_401360(&SuccessFlag);
16  sub_401200(&WaFlag);
17  if ( InputCheck1(Input, (int)&WaFlag) )       // 判断输入中是否均为数字和字母
18  {
19    if ( &Input[strlen(Input) + 1] - &Input[1] == 22 )// len(Input) == 22
20      sub_401C40(Input, (int)&SuccessFlag, (int)&WaFlag);
```


401C40函数：

```
Init(&v11, &Src);
AddItemInTrieTree(&Root, (int)&v11);          // flag[13:16]
Init(&v10, &v41);
AddItemInTrieTree(&Root, (int)&v10);          // flag[0:2]
Init(&v9, &v47);
AddItemInTrieTree(&Root, (int)&v9);           // flag[9:13]
Init(&v8, &v24);
AddItemInTrieTree(&Root, (int)&v8);           // flag[4:7]
Init(&v7, &v28);
AddItemInTrieTree(&Root, (int)&v7);           // flag[2:4]
Init(&v6, &v44);
AddItemInTrieTree(&Root, (int)&v6);           // flag[7:9]
Init(&v5, &v35);
AddItemInTrieTree(&Root, (int)&v5);           // flag[16:19]
Init(&v4, &v20);
AddItemInTrieTree(&Root, (int)&v4);           // flag[19:22]
if ( sub_4030E0(&Root, (int)&dword_407E48) )
  XORCalc(&v41, &v44, (int)&Src, (int)&v35, SuccessFlag, WaFlag);
else
  Printf(WaFlag);
```


XORCalc函数（401B80函数）：

```
1 int __cdecl sub_401B80(char *a1, char *a2, int a3, int a4, int a5, int a6)
2 {
3   int result; // eax
4
5   if ( (a1[1] ^ *a1) != 84
6     || (a2[1] ^ *a2) != 19
7     || (*(char *)(a3 + 1) ^ *(char *)(a3 + 2)) != 18
8     || (*(char *)(a4 + 2) ^ *(char *)(a4 + 1)) != 77 )
9   {
10    result = Printf(a6);
11  }
12  else
13  {
14    result = Printf(a5);
15  }
16  return result;
17}
```

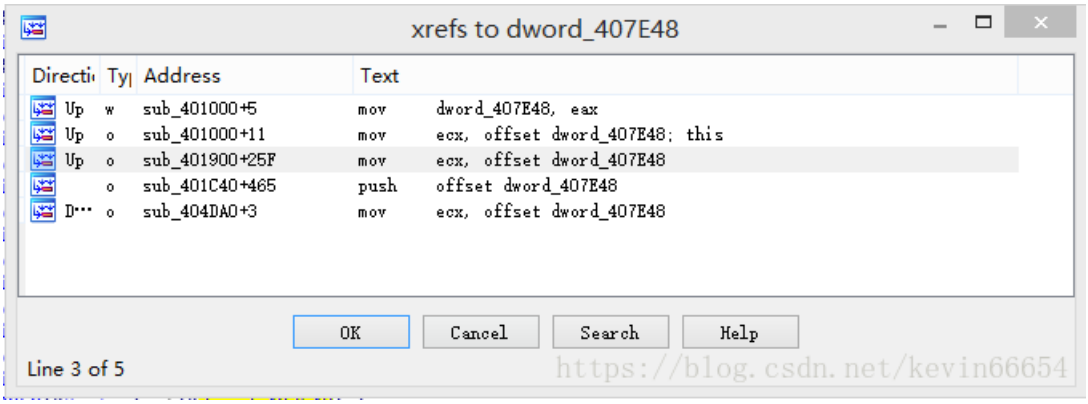

给出了4个异或关系用来限制输入的flag：

s[0] ^ s[1] = 84，s[7] ^ s[8] = 19，s[14] ^ s[15] = 18，s[17] ^ s[18] = 77

所以重点应该放在dword_407E48s上

```
111  if ( sub_4030E0(&Root, (int)&dword_407E48) )
112    XORCalc(&v41, &v44, (int)&Src, (int)&v35, SuccessFlag, WaFlag);
```

使用JumpToXref功能找到关键函数：



根据题目（数据结构）以及程序中字符串

```
's' .data:004··· 0000000F C .?AVTrieTree@@
's' .data:004··· 00000012 C .?AVTrieTreeAbs@@
's' .data:004··· 00000013 C .?AVTrieTreeNode@@
's' .data:004··· 00000016 C .?AVTrieTreeNodeAbs@@
's' .data:004··· 00000019 C .?AVTrieTreeNodeStatic@@
's' .data:004··· 00000015 C .?AVTrieTreeStatic@@
```

猜测这个题是字典树的实现，以及字符串的添加与匹配

那么思路就有了：判断的是我们输入的flag，在分段添加进入字典树之后，与原有的字典树匹配

IDA找其构造的函数，401900函数：

```
21  v10 = this;
22  sub_4015D0(&v12);                           // f
23  sub_401540(&v13);                           // t
24  sub_4016E0(&v17);                           // M
25  sub_401660(&v14);                           // 7
26  sub_401880(&Src);                           // 9
27  sub_401770(&v16);                           // k
28  sub_4014C0(&v15);                           // c
29  sub_4017F0(&v11);                           // kx
```

```
0099196D  .  51          push ecx
0099196E  .  E8 7DFEFFFF call 2018CMv4.009917F0
00991973  .  83C4 04     add esp, 0x4
00991976  .  8D55 F8     lea edx, [local.2]
```

```
EAX 00000078
ECX 00BCF8F8
EDX 00000002
```

```
00BCF4D0 : 00BCF8F8  ASCII "kx"
```

一步一步调试可以得到这些中间值

这一段的过程是：在内存中生成这些字符串

```
30  Init(&v9, &Src);
31  fill_node_string((int)&unk_4074B0, &v9);      // 9
32  Init(&v8, &v17);
33  fill_node_string((int)&unk_4075C0, &v8);      // M
34  Init(&v7, &v16);
35  fill_node_string((int)&unk_407D38, &v7);      // k
36  Init(&v6, &v15);
37  fill_node_string((int)&unk_4077E8, &v6);      // c
38  Init(&v5, &v14);
39  fill_node_string((int)&unk_407A08, &v5);      // 7
40  Init(&v4, &v13);
41  fill_node_string((int)&unk_407C28, &v4);      // t
42  Init(&v3, &v11);
43  fill_node_string((int)&unk_4076D0, &v3);      // kx
44  Init(&v2, &v12);
45  fill_node_string((int)&unk_407B18, &v2);      // f
```


逻辑是：将字符串转为节点

```
46  add_child_node(&unk_4075C0, (int)&unk_407D38);
47  add_child_node(&unk_407C28, (int)&unk_4074B0);
48  add_child_node(&unk_407A08, (int)&unk_4075C0);
49  add_child_node(&unk_407C28, (int)&unk_407B18);
50  add_child_node(&unk_4077E8, (int)&unk_407A08);
51  add_child_node(&unk_4078F8, (int)&unk_4076D0);
52  add_child_node(&unk_4077E8, (int)&unk_407C28);
53  add_child_node(&unk_4078F8, (int)&unk_4077E8);
54  set_occurence(&unk_4077E8, 0);
55  set_occurence(&unk_407D38, 1);
56  set_occurence(&unk_4074B0, 1);
57  set_occurence(&unk_407C28, 1);
58  set_occurence(&unk_407A08, 1);
59  set_occurence(&unk_4075C0, 2);
60  set_occurence(&unk_4078F8, 0);
61  set_occurence(&unk_407B18, 1);
62  set_occurence(&unk_4076D0, 1);
63  sub_403A60(&dword_407E48, (int)&unk_4078F8);
64  return v10;
```

需要的是407E48，来自于4078F8，相当于字典树的根节点：

那么就需要正向构造这棵树：根据IDA逻辑手动构造：

```
46  add_child_node(&unk_4075C0, (int)&unk_407D38);// M -> k
47  add_child_node(&unk_407C28, (int)&unk_4074B0);// t -> 9
48  add_child_node(&unk_407A08, (int)&unk_4075C0);// 7 -> M
49  add_child_node(&unk_407C28, (int)&unk_407B18);// t -> f
50  add_child_node(&unk_4077E8, (int)&unk_407A08);// c -> 7
51  add_child_node(&unk_4078F8, (int)&unk_4076D0);// root -> kx
52  add_child_node(&unk_4077E8, (int)&unk_407C28);// c -> t
53  add_child_node(&unk_4078F8, (int)&unk_4077E8);// root -> c
54  set_occurence(&unk_4077E8, 0);                // c出现0次
55  set_occurence(&unk_407D38, 1);                // c7Mk出现1次
56  set_occurence(&unk_4074B0, 1);                // ct9出现1次
57  set_occurence(&unk_407C28, 1);                // ct出现1次
58  set_occurence(&unk_407A08, 1);                // c7出现1次
59  set_occurence(&unk_4075C0, 2);                // c7M出现2次
60  set_occurence(&unk_4078F8, 0);
61  set_occurence(&unk_407B18, 1);                // ctf出现1次
62  set_occurence(&unk_4076D0, 1);                // kx出现1次
63  sub_403A60(&dword_407E48, (int)&unk_4078F8);
```


这就是flag中出现的字符，且需要满足异或的关系

根据长度，flag[9:13]长度为4，为c7Mk

```
import string
ch = string.letters + string.digits

t = []
for i in ch:
 for j in ch:
  if ord(i) ^ ord(j) == 84:
   t.append(i + j)
print "input[0:2] :",t

t = []
for i in ch:
 for j in ch:
  if ord(i) ^ ord(j) == 19:
   t.append(i + j)
print "input[7:9] :",t

t = []
for i in ch:
 for j in ch:
  if ord(i) ^ ord(j) == 18:
   t.append(i + j)
print "input[14:16] :",t

t = []
for i in ch:
 for j in ch:
  if ord(i) ^ ord(j) == 77:
   t.append(i + j)
print "input[17:19] :",t
```

得到flag[0:2]为c7，flag[16:19]为ct9，flag[7:9]为kx，flag[14:16]为tf

长度为2的只剩一个了，得flag[2:4]为ct

连起来：c7ctc7Mkxc7Mkctfct9c7M

PLUS：

原作者的出题思路以及CrackMe源代码

https://bbs.pediy.com/thread-222959.htm

使用strings查看CM，发现：